How to build web self-service by functional profiles?

Benjamin Chevallereau
Ecole Centrale de Nantes
IRCCyN
Nantes, France
benjamin.chevallereau@irccyn.ec-nantes.fr

Alain Bernard
Ecole Centrale de Nantes
IRCCyN
Nantes, France
mailto:alain.bernard@irccyn.ec-nantes.fr

***Abstract*—Companies, institutions, and local authorities look for virtualizing all provided services to obtain more profit in terms of productivity, profitability, quality of service, traceability… The main issues are the identification of problems to solve and the choice of adapted solutions. To build these ones, functional and technical profiles must communicate to understand precisely the domain, the problem, and the expected features… This paper provides a framework composed of three contributions which aims at simplifying the building of web self-service by functional profiles. The first part deals with a language to specify functional needs using the main concept of goal. This language must be as simple as possible but not limiting. The second one is the tool to use this language and, finally, the last section is a set of interpretations to ease the communication between all involved people. To automate a great part of recurrent tasks, all the proposed approach complies with model driven methodology.**

***Index Terms*—Generation, model driven engineering, requirements engineering, web self-service.**

I. What is web self-service?

WEB self-service is a category of electronic support that allows customers, employees and citizens to access information and perform routine tasks over the Internet, without requiring any direct interaction with a representative of an enterprise. Web self-service is widely used in customer relationship management (CRM), employee relationship management (ERM) and especially in e-government.

When it is specific to Web-enabled employee interactions, the practice is known as employee self-service (ESS). When it is specific to Web-enabled customers, it is called customer self-service (CSS). For employees and customers, self-service offers 24 hour-a-day support, and immediate access to information without having to wait for an email response or a returned telephone call. Ultimately the success of Web self-service is based on the quality and the quantity of information available and the ease with which it can be accessed.

Deploying Web self-service applications results in benefits for company by many ways. The most prominent motivation is a lower cost, as compared with telephone or email service by a company representative.

Yet such projects typically involve web content management (WCM), search, and enterprise content management (ECM) technologies. None of these technologies are simple to procure, develop, implement, or run. And if you get things wrong, then self-service and automation can backfire, destroying customer loyalty, and boosting exception-handling costs, rather than reducing expenses.

Defining a web self-service is typically a task that must be carry out by functional profile and more precisely by customers. This definition is therefore used to write a technical specification usable by developers to build the system. This translation is a very important step and can generate troubles.

II. Toolbox to build web self-service

The proposed approach aims to overcome communication problems between technical and functional experts by organizing needs specification in a common language and by providing a mechanism to improve information exchange between stakeholders (Fig. 1). It is composed of (a) a metamodel or a language for functional needs, (b) a modeler adapted to functional experts and finally (c) a set of automatic interpretation, i.e. model transformation, generation and post-processing task.

A. Language

The language is composed of three main concepts and five main links. The definition of a web-self service is represented as a goal tree. Next, this tree is completed by a set of agents representing future user profiles and a set of entities defining data managed during the web self-service. Entities must be linked between them by relationship. Each agent must be responsible of a non-empty set of goals and each goal is linked to data by a strategy. It represents necessary rights on the information system to achieve this goal. The last link deals with the sequentiality between goals, by defining which goal must be achieved before another one.

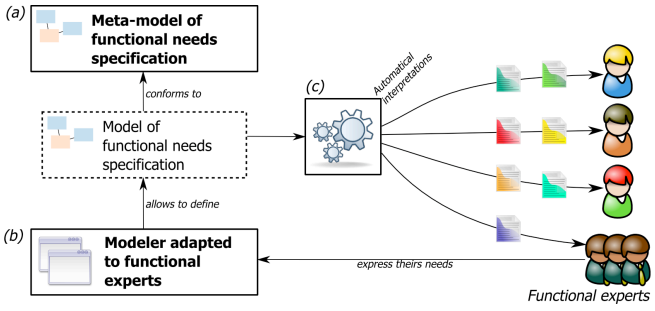

Fig. 1. Global schema of our proposition.

B. Modeler

A graphical tool was built to use this language. This one

978-1-4244-6273-5/09/$26.00 ©2009 IEEE

used the toolkit defined in the TOPCASED[1] project and was customized so as to be used by functional profiles. However this first version of the tool provides a too technical environment (Fig. 2).

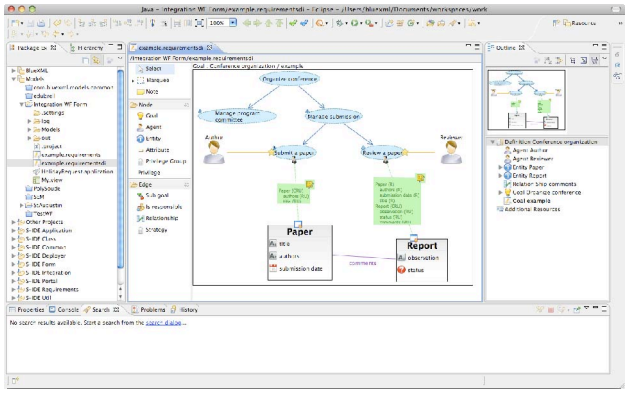

Fig. 2. Modeler provided to use the language.

It is why a more adapted and user-friendly tool was created. This modeler is usable through a traditional web browser (Fig. 3). Thus a functional expert can easily define its required web self-service. Moreover the model can be retrieved and exported towards a technical environment for developers.

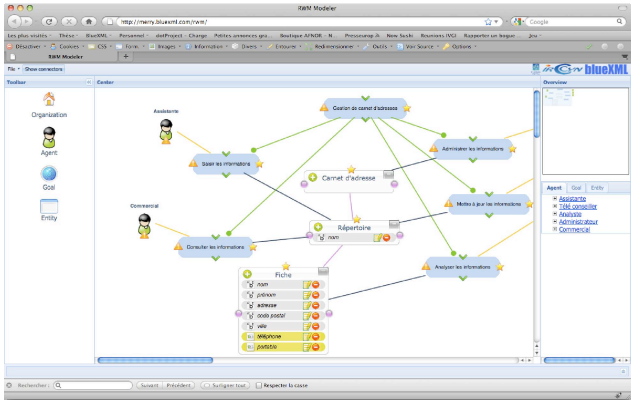

Fig. 3. Web modeler provided to use the language.

C. *Interpretation*

In order to improve the reliability of the communication process, the automation of interpretations is proposed. These automated interpretations aim at increasing the quality of needs specification by reducing as much as possible human misunderstanding. Moreover, they reduce the time spent to explain, illustrate and communicate the specification among the different stakeholders. Indeed when many stakeholders are involved, that can cost. Of course, it is not possible to use a unique communication support with all stakeholders. It is why the modeler proposes interpretations of needs specification to different domains using different visualizations.

The mechanism of interpretation is decomposed in three steps (Fig. 4). The source of an interpretation is a model of needs specification matching with the language defined previously. The first step is a model transformation using ATL (ATLAS Transformation Language)[1]. This language is a hybrid model transformation language that allows both declarative and imperative constructs to be used in transformation definitions. The target metamodel is specific to the interpretation. The next step is the generation step. This is achieved with generation templates using Acceleo[2]. Finally, the last step executes a post-processing using ANT scripts. An interpretation can be represented as a triplet composed of a transformation model linked with the target metamodel, a set of generation templates and a set of scripts.

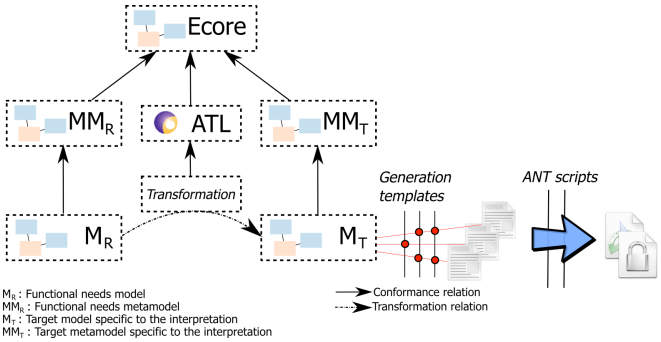

Fig. 4. Interpretation mechanism.

III. EXAMPLE

We can see on the figure 5 a partial needs specification for a conference organization system. The current example is focused on the paper submission process. First the goals and the agents are defined and modeled. The goal "*Manage submissions*" is composed of two sub-goals: "*Submit a paper*" under the responsibility of the author and "*Review a paper*" under the responsibility of the reviewer. The reviewer and the author are represented with the concept of agents defined as a

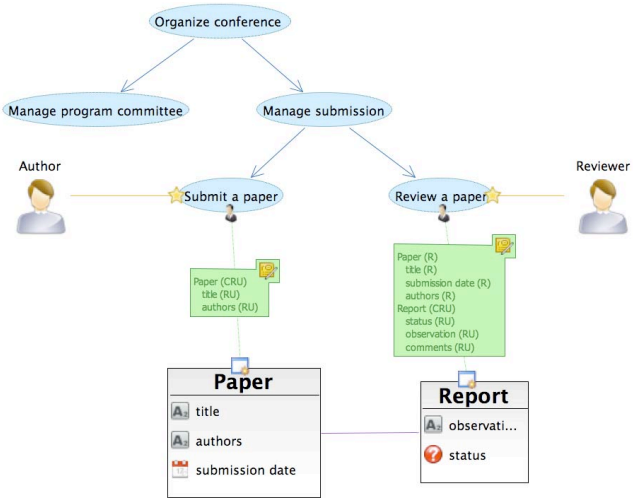

Fig. 5. Simple part of an example for a conference organization.

[1] TOPCASED project aims to provide a development toolkit dedicated to critical and embedded systems, software and hardware. The subproject used is TOPCASED-MF, which is the modeling framework of the TOPCASED project. http://www.topcased.org/

future user profile in the new software. Then information around the objects involved by these goals is gathered and modeled, with some entities, some relations to link them and some attributes to characterize them. The figure shows two entities: "*Paper*" and "*Report*". After defining the goals and the objects, it is possible to link them with the concept of strategy. In order to meet functional needs of each agent, we can define a path in the information structure for each goal under agents' responsibility composed of a set of privileges. A path can be seen as a partial view on the future information system. The paper submission has a simple path: the author enters in the software by the concept of "*Paper*" and can create one or many instances. The reviewing path of an article is slightly more complex: the reviewer enters in the software by the concept of "*Report*" and can navigate in the software to select the article concerned by his review.

A. Model transformation

This model can be interpreted to get models adapted to the different stakeholders (developers, users, managers, customers…). In this example, we demonstrate an interpretation to get a set of conceptual model. The first deduced model is the workflow model (Fig. 6). If we want to implement a process on an information system, we must define a workflow. This workflow conforms to the notation *jPDL*[2]. Each process defined in the requirements model can be transformed as workflow model.

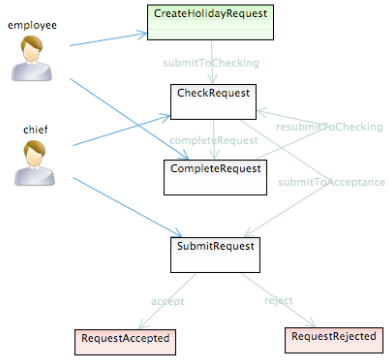

Fig. 6. Workflow model.

From the entity-relationship part is deduced a data model (Fig. 7). This model supports a detailed design process. Each entity is transformed into a class and each relationship into an association.

Once the data and workflow models are obtained, form models can be deduced to make the link between them. Each user task in the workflow model is connected to a form linked to data.

B. Generation

The next step is the generation. Software can be entirely generated. The data model is used to configure an ECM platform like *Alfresco*. The workflow model is used to configure the *jBoss jBPM* and forms model are used to generate *XForms*. All components are technically linked and can be used directly to test and to integrate.

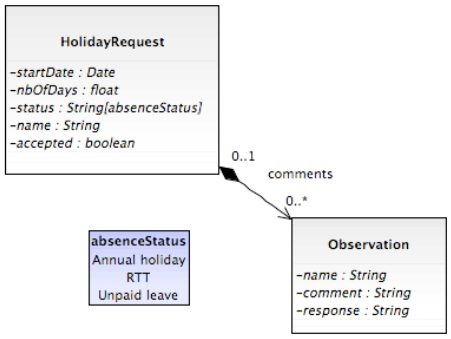

Fig. 7. Class model.

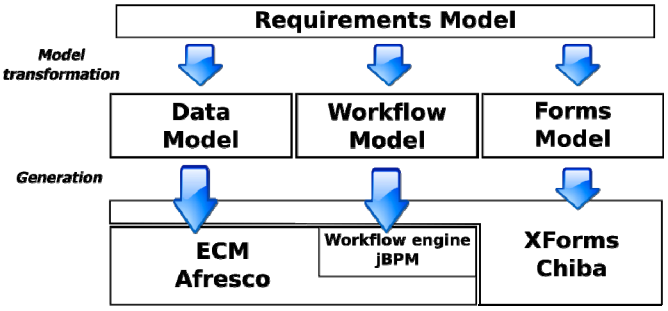

Fig. 8. Generation step.

IV. Conclusion

This paper proposes some tools to ease the design and the development of software. This example is focused on the modeling and the generation of web self-service. The two main goals are to improve the communication between experts and to make more productivity profits. The first one is solved by a proposition of simple language and the set of interpretations allowing transforming a model using this language to another technological space. We use model driven environment to give an answer to the second goal. All translations, transformations, generations… used concepts including in MDE.

References

[1] F. Jouault and I. Kurtev, "Transforming Models with ATL, ", satellite events at the MoDELS 2005 Conference, pp. 128–138, 2006.

[2] Obeo: Acceleo Generator. http://www.acceleo.org

[2] jBPM Process Definition Language : http://docs.jboss.org/jbpm/v3/userguide/jpdl.html